Using Deception in Markov Game to Understand Adversarial Behaviors through a Capture-The-Flag Environment





Siddhant Bhambri[1]* Purv Chauhan[1]* Frederico Araujo[2] Adam Doupé[1] Subbarao Kambhampati[1]

[1]Arizona State University, [2]IBM Research

Objective

- Identifying real-world adversarial threats
- Advantage of deploying deception strategies
- Modeling the real-world attacker
- Role of game-theoretic decision models
- Understanding human attacker behaviors

Agenda

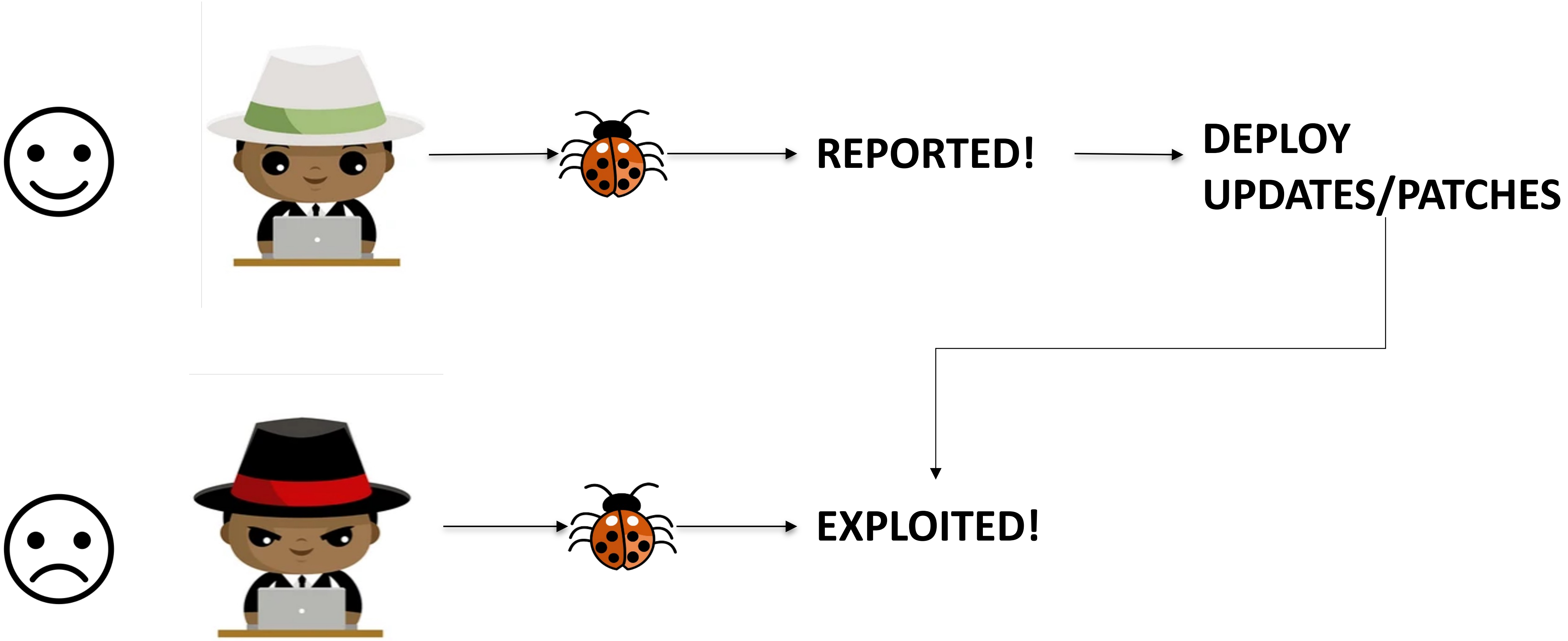
- Reviewing Honey-Patching
- Setting up the user-study
- Deploying Honey-Patches as mitigations
- Modeling the adversary’s attack graph
- Formulating the Markov Game model
- Computing the Bayesian Stackelberg Equilibrium

Traditional Solution to Mitigate Attacks

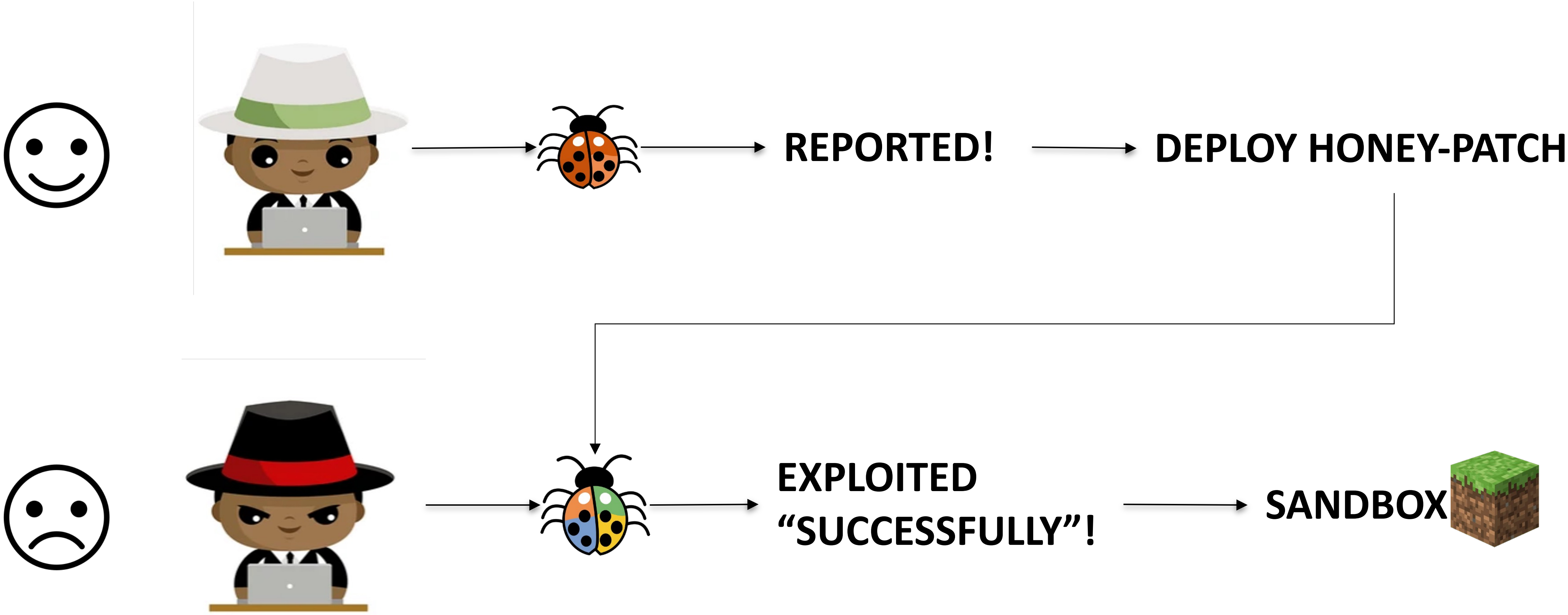
REPORTED!

DEPLOY UPDATES/PATCHES

Traditional Solution to Mitigate Attacks



Using Deception as Defense

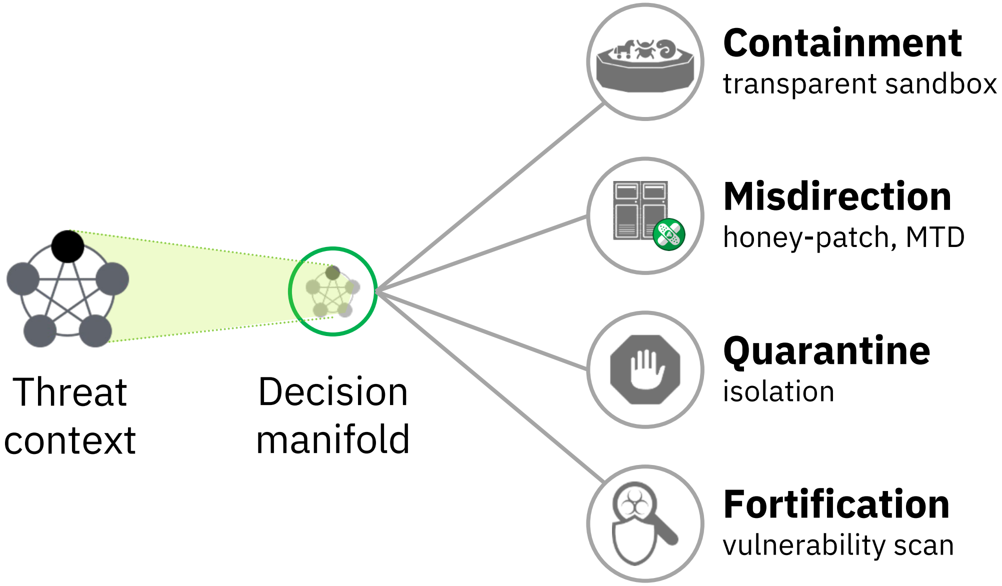


Araujo, Frederico, and Teryl Taylor. "Improving cybersecurity hygiene through JIT patching." *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2020.

Agenda

- Reviewing Honey-Patching
- Setting up the user-study
- Deploying Honey-Patches as mitigations
- Modeling the adversary's attack graph
- Formulating the Markov Game model
- Computing the Bayesian Stackelberg Equilibrium

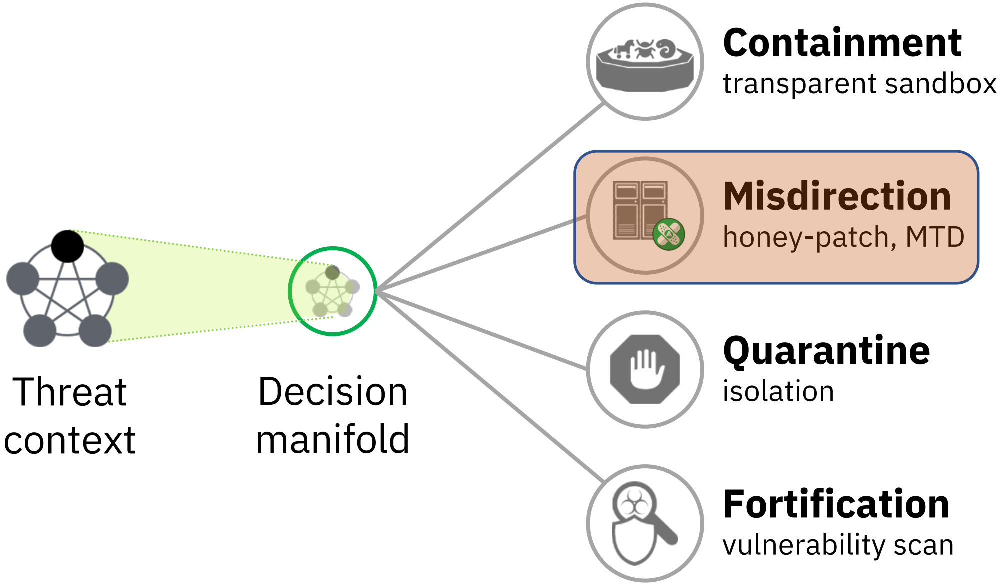
Automating Proactive Responses



Which one to choose?

I want to **disrupt attack kill chains** and perform **attacker reconnaissance**!

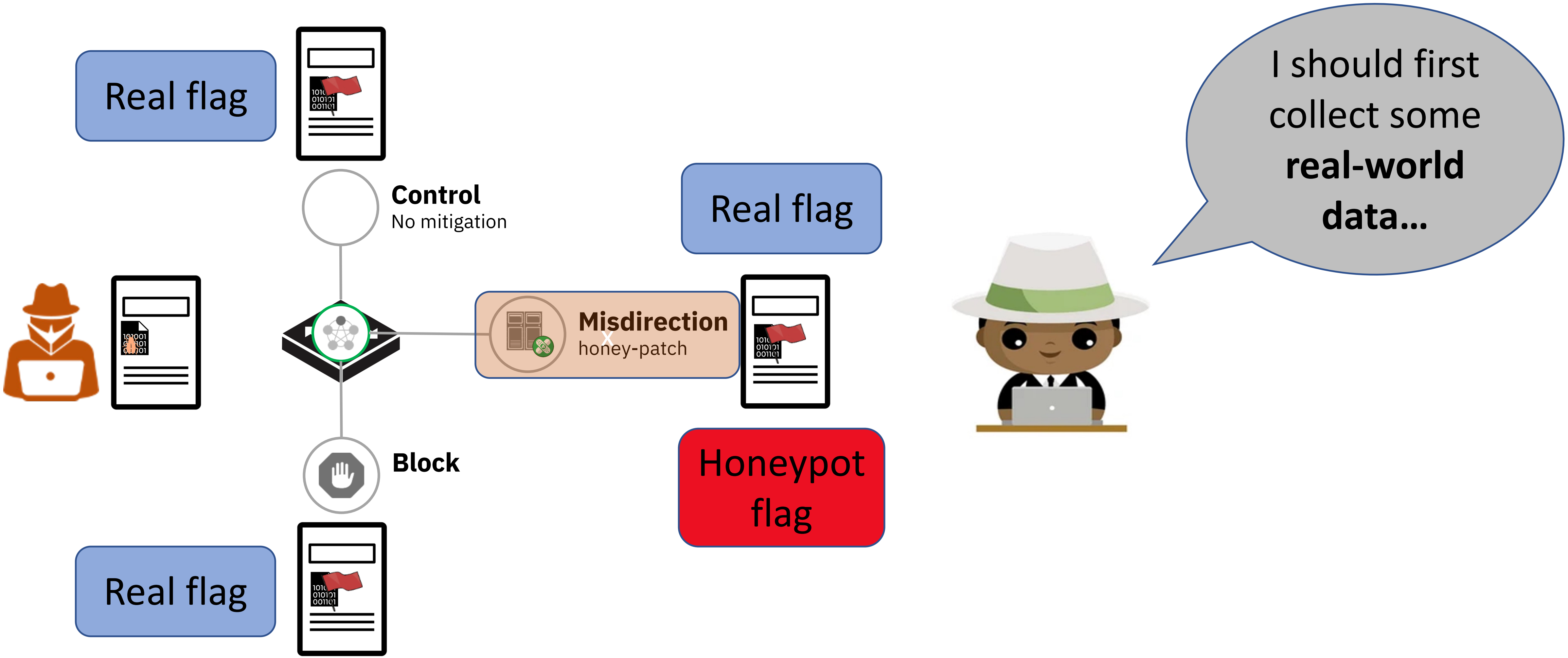
Automating Proactive Responses



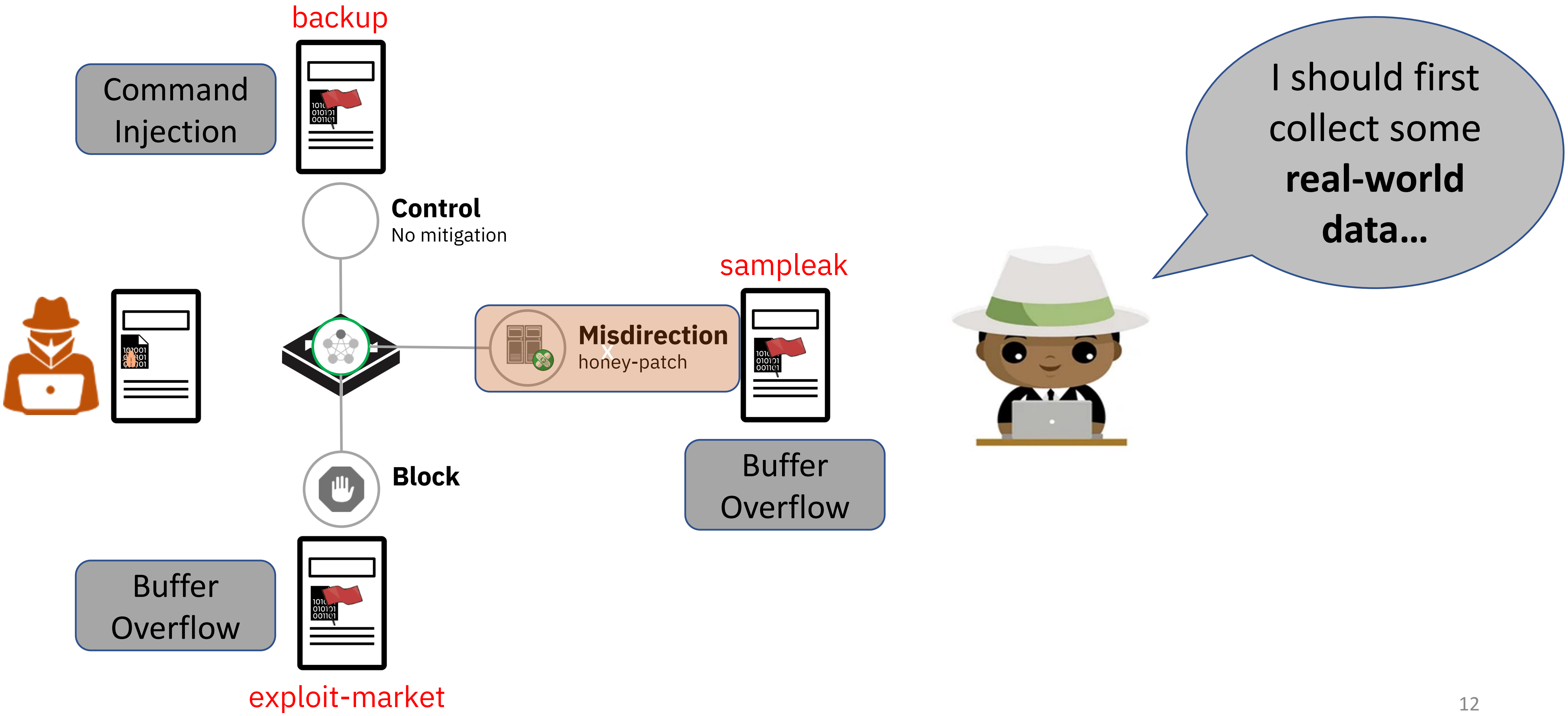
I also need to understand the **attacker's behavior**!

Heckman, Kristin E., et al. "Cyber denial, deception and counter deception." *Advances in Information Security* 64 (2015).
Araujo, Frederico, et al. "From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation." *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*. 2014.

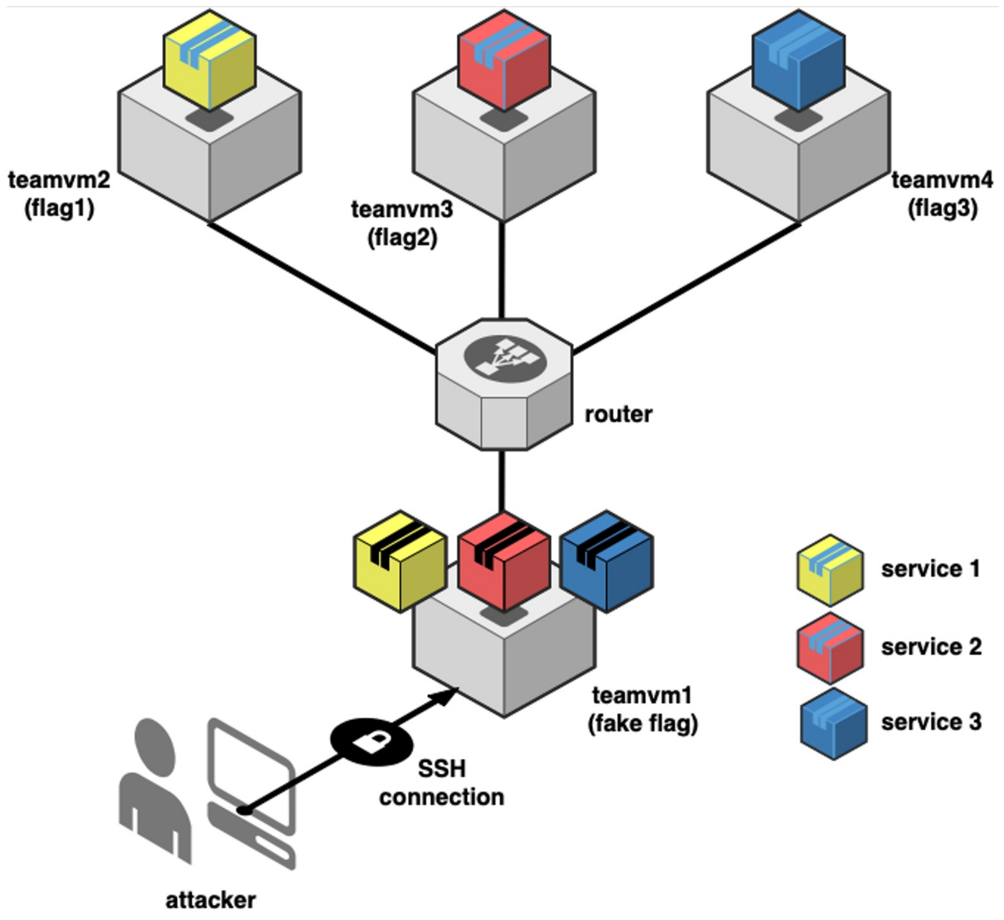
Testing Deception as Defense Strategy

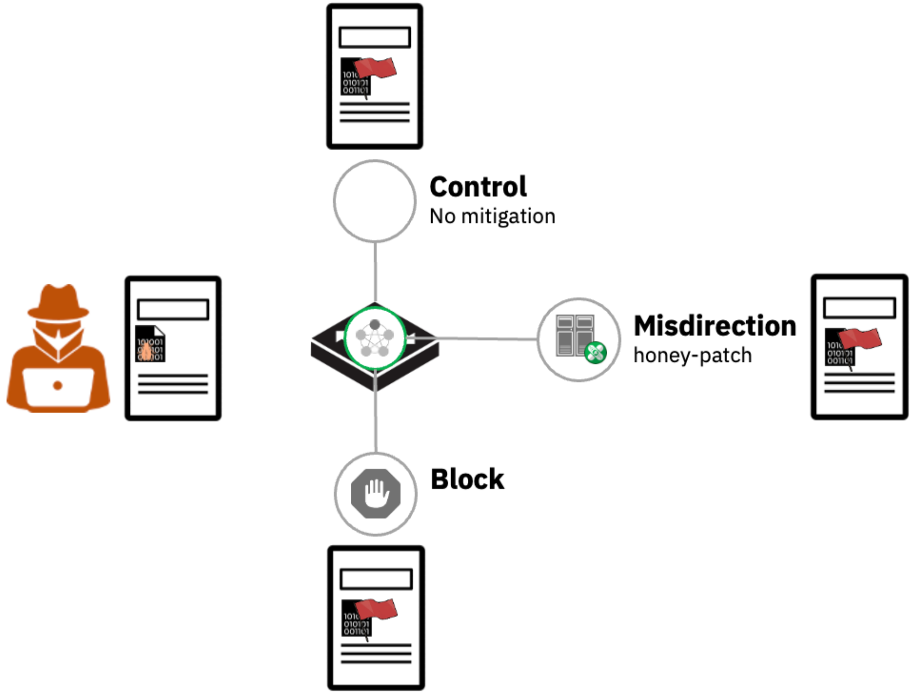


Testing Deception as Defense Strategy



Adapting the iCTF Framework

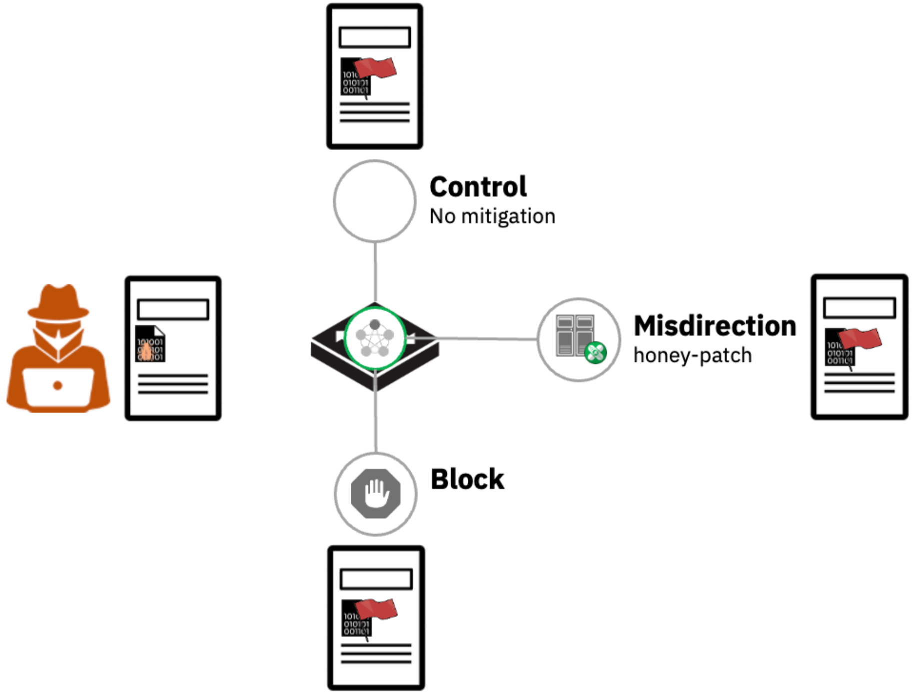




Vigna, Giovanni, et al. "Ten Years of {iCTF}: The Good, The Bad, and The Ugly." *2014 USENIX Summit on Gaming, Games, and Gamification in Security Education (3GSE 14)*. 2014.

Hypothesis Testing

"Once trapped in a honeypot environment, the attacker *chooses to continue* with the existing strategy to exploit the remaining vulnerabilities."

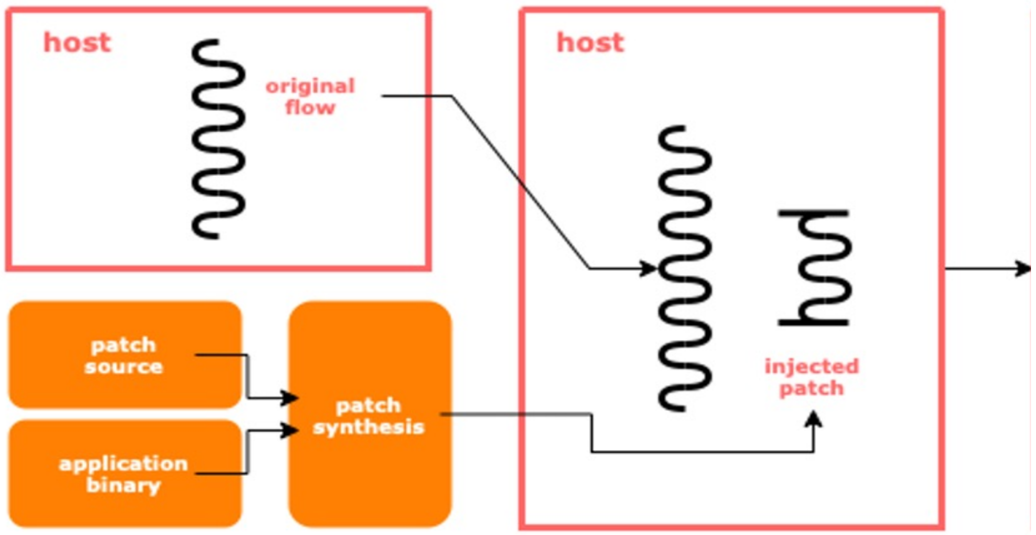


Agenda

- Reviewing Honey-Patching
- Setting up the user-study
- Deploying Honey-Patches as mitigations
- Modeling the adversary’s attack graph
- Formulating the Markov Game model
- Computing the Bayesian Stackelberg Equilibrium

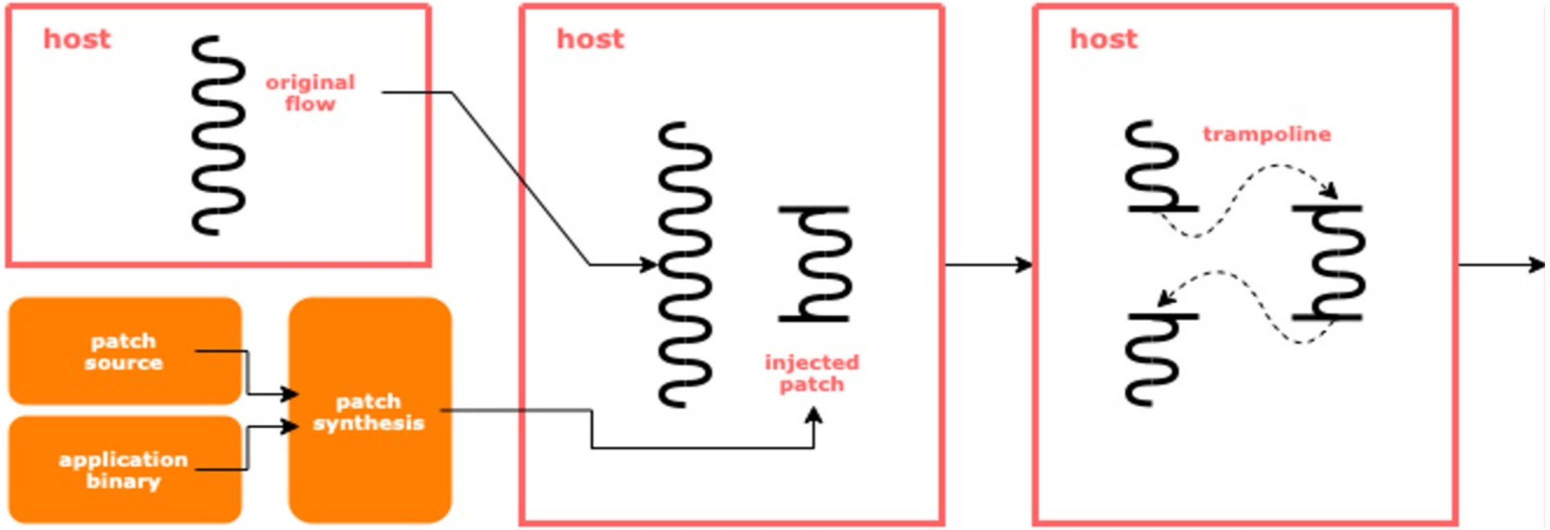
Deploying the Honey-Patching Framework

1. Patch synthesis 2. Inject patch and compile



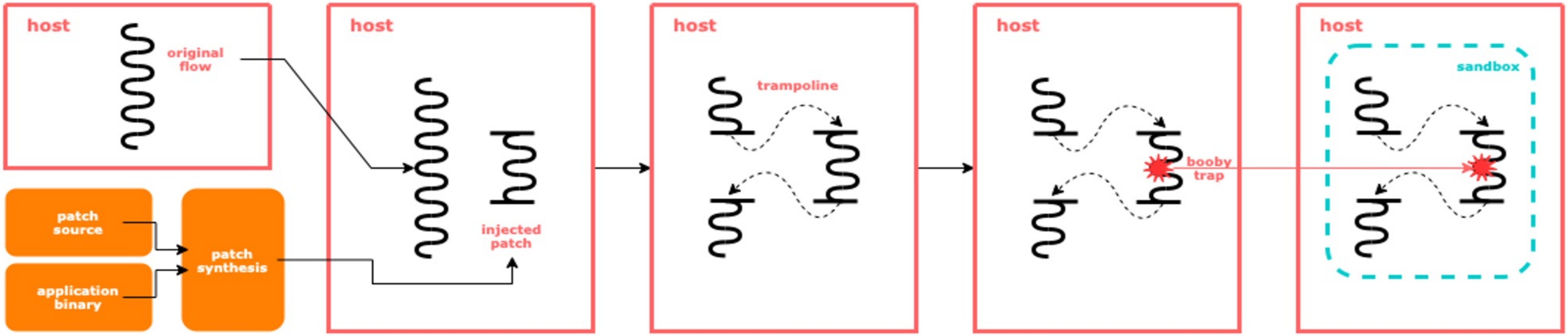
Deploying the Honey-Patching Framework

1. Patch synthesis   2. Inject patch and compile   3. Setup trampolines



Deploying the Honey-Patching Framework

1. Patch synthesis
2. Inject patch and compile
3. Setup trampolines
4. Booby trap triggered
5. Sandbox attacker



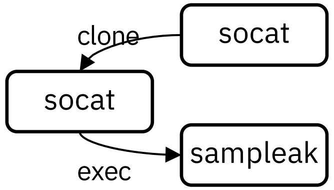
INSIDER Honey-Patching Framework

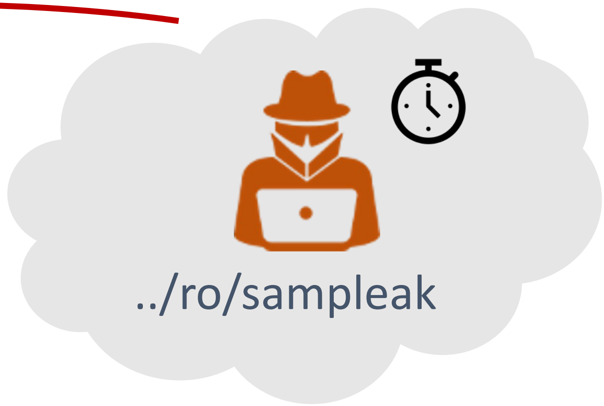
Araujo, Frederico, and Teryl Taylor. "Improving cybersecurity hygiene through JIT patching." *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2020.

Deploying the Honey-Patching Framework

Original Chall

```
FROM ubuntu:18.04
...
CMD cd service/rw && socat tcp-l:6666,reuseaddr,fork
exec:"../ro/sampleak"
```

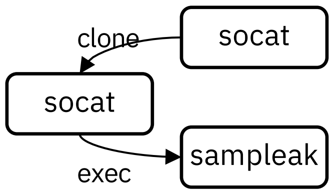


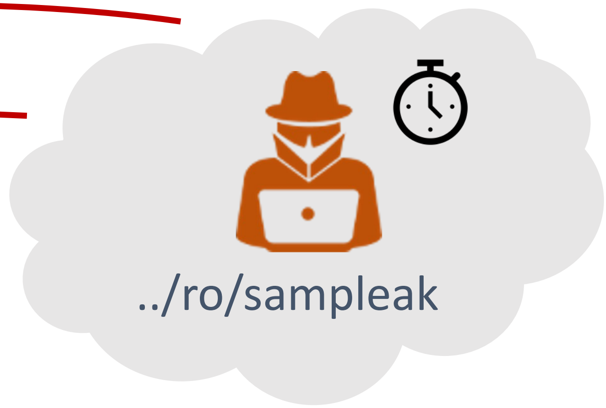


Deploying the Honey-Patching Framework

Original Chall

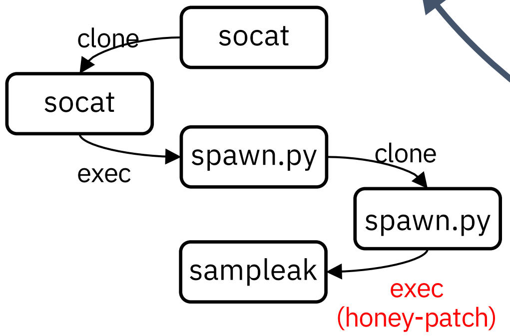
```
FROM ubuntu:18.04
...
CMD cd service/rw && socat tcp-l:6666,reuseaddr,fork
exec:"../ro/sampleak"
```





Honey-Patched Chall

```
FROM ubuntu:18.04
...
CMD cd service/rw && socat tcp-l:6666,reuseaddr,fork
exec:"/insider/spawn.py patch.conf ../ro/sampleak"
```

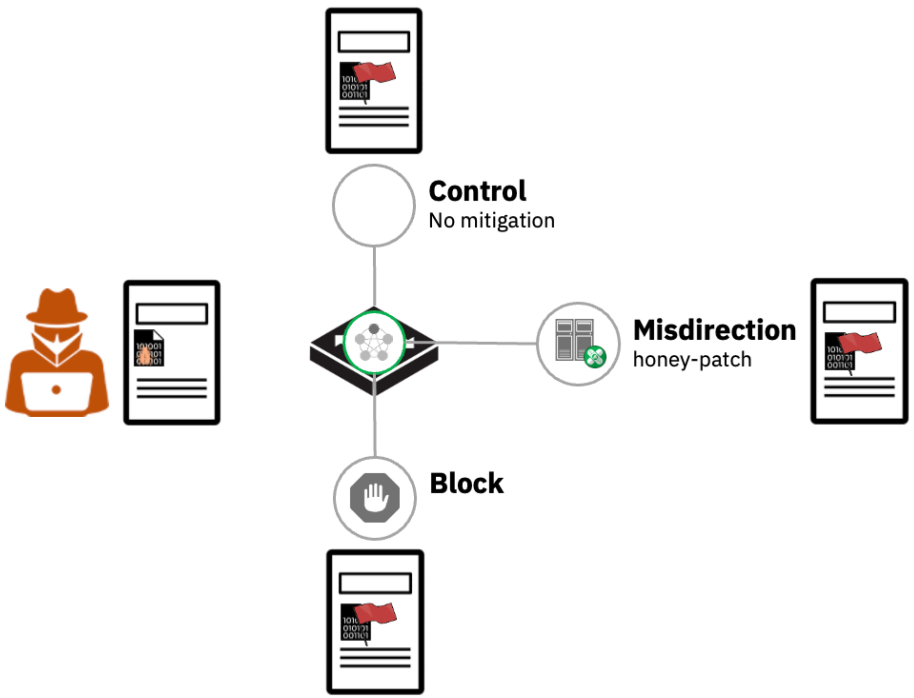


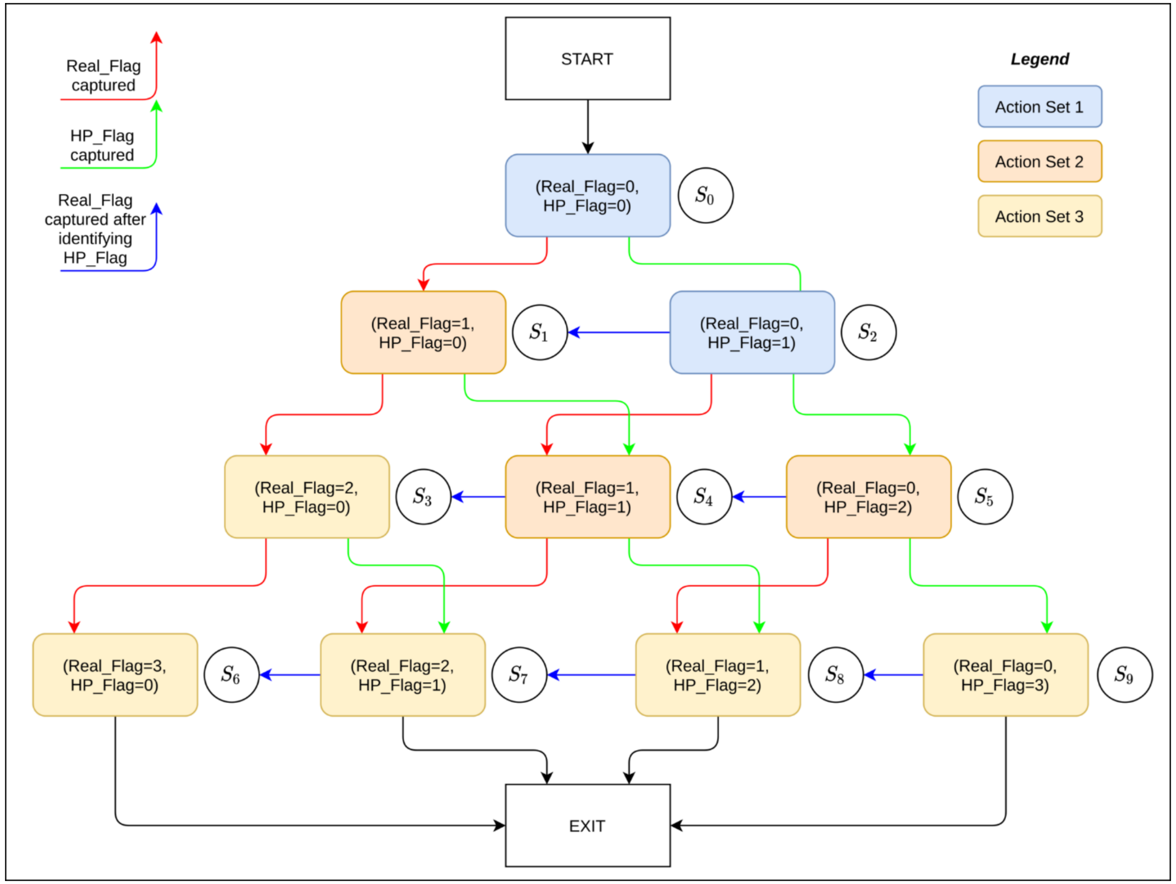

```
{
    "__type__": "config",
    "agentscript": "patch.js",
    "spawnscript": "spawn.js",
    "elfpaths": [
        {
            "path": "/insider/frida/examples/sampleak/sampleak",
            "pic": false
        }
    ],
    "bcpath": "patch.ll",
    "replacements": [
        {
            "targetfn": "is_easy_password",
            "surrogatefn": "_p_is_easy_password",
            "fntype": { "rettype": "int", "argstype": ["pointer"] }
        }
    ],
    "logpath": "/insider/frida/examples/sampleak/spawn_logs",
    "target": "sampleak"
}
```

Agenda

- Reviewing Honey-Patching
- Setting up the user-study
- Deploying Honey-Patches as mitigations
- Modeling the adversary's attack graph
- Formulating the Markov Game model
- Computing the Bayesian Stackelberg Equilibrium

Attack Graph

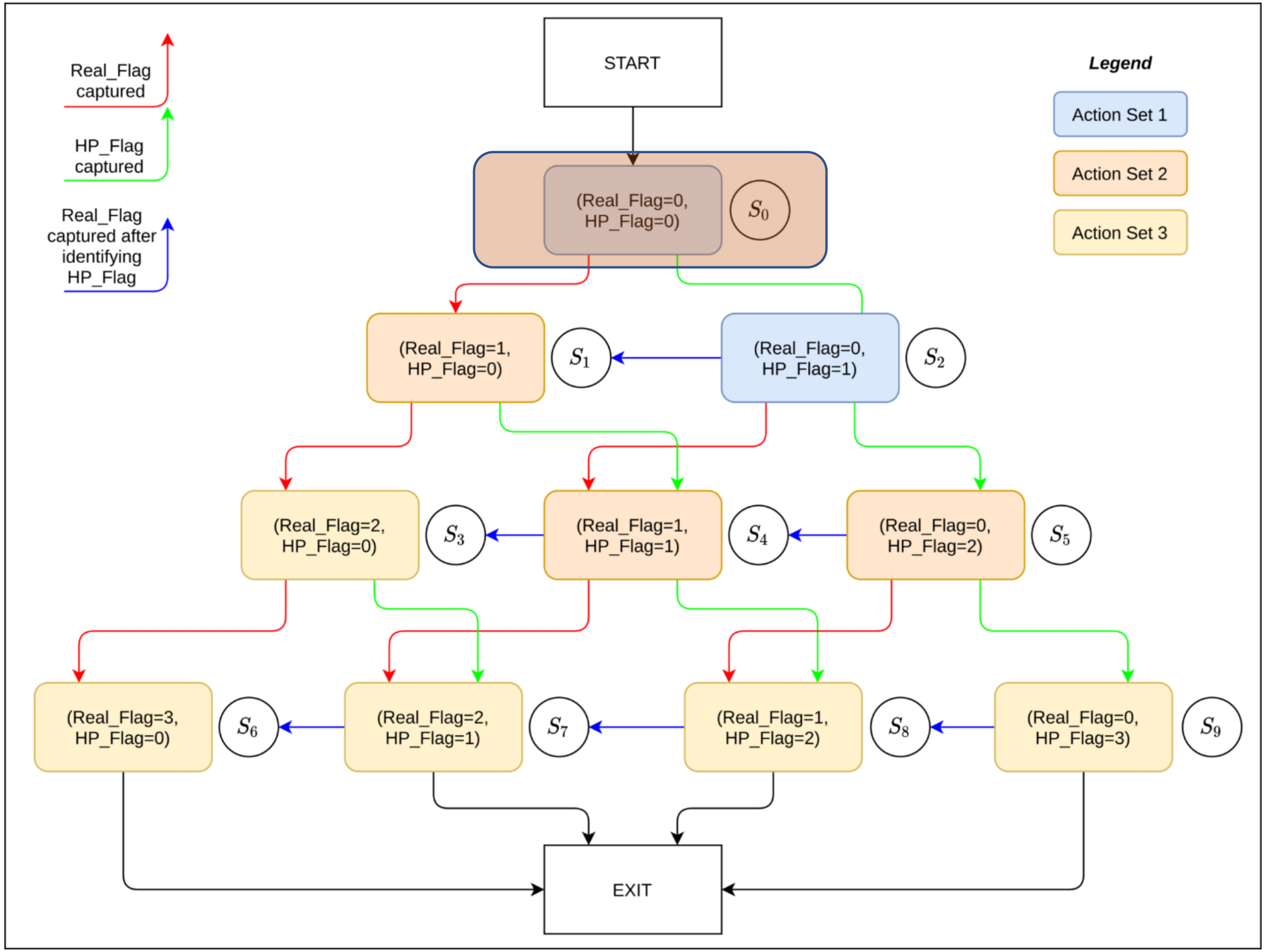




Lallie, Harjinder Singh, Kurt Debattista, and Jay Bal. "A review of attack graph and attack tree visual syntax in cyber security." *Computer Science Review* 35 (2020): 100219.
Winterrose, Michael L., et al. "Adaptive attacker strategy development against moving target cyber defenses." *Advances in Cyber Security Analytics and Decision Systems*. Springer, Cham, 2020.

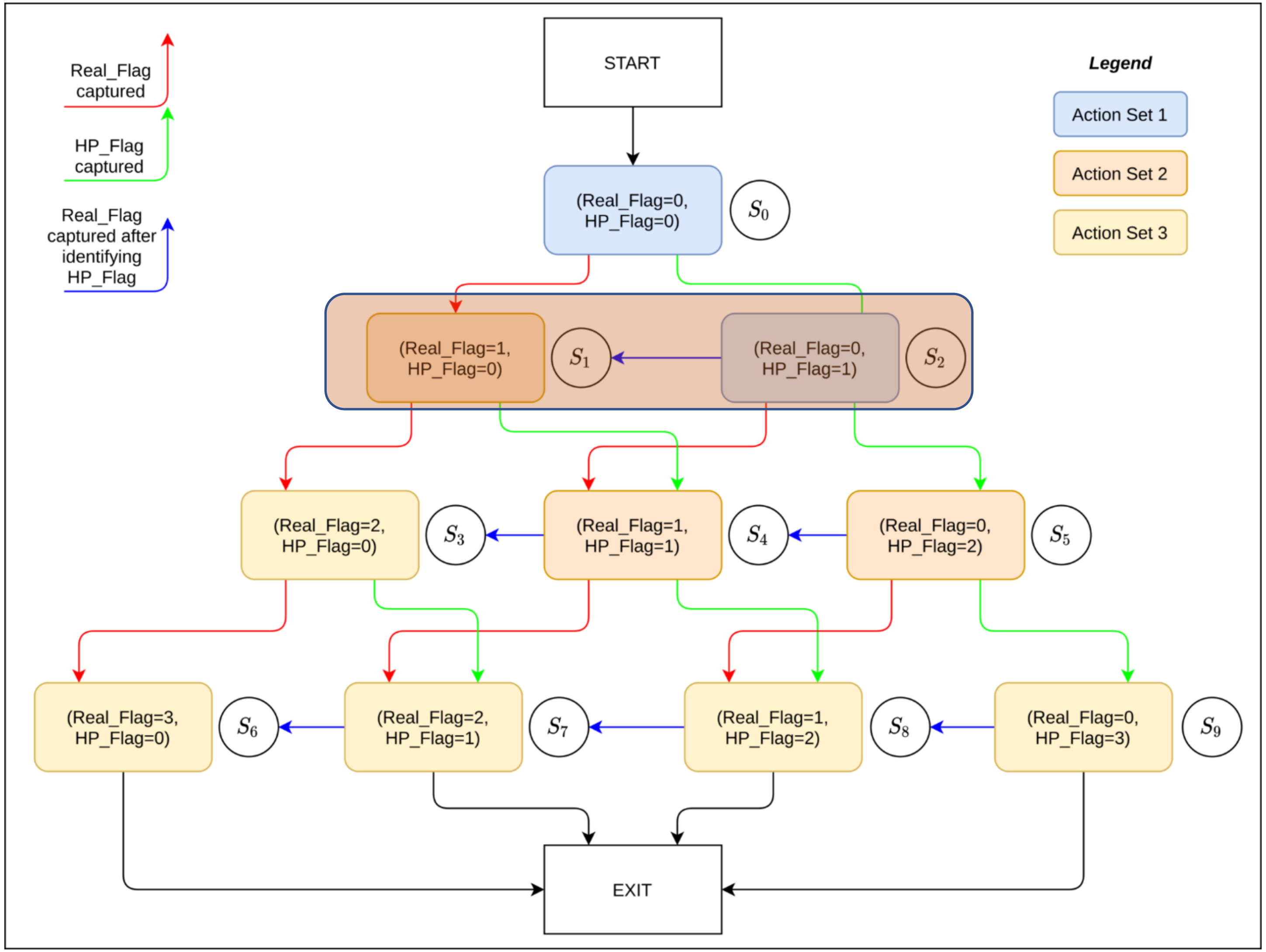
Attack Graph

Let me **start the exploit**!

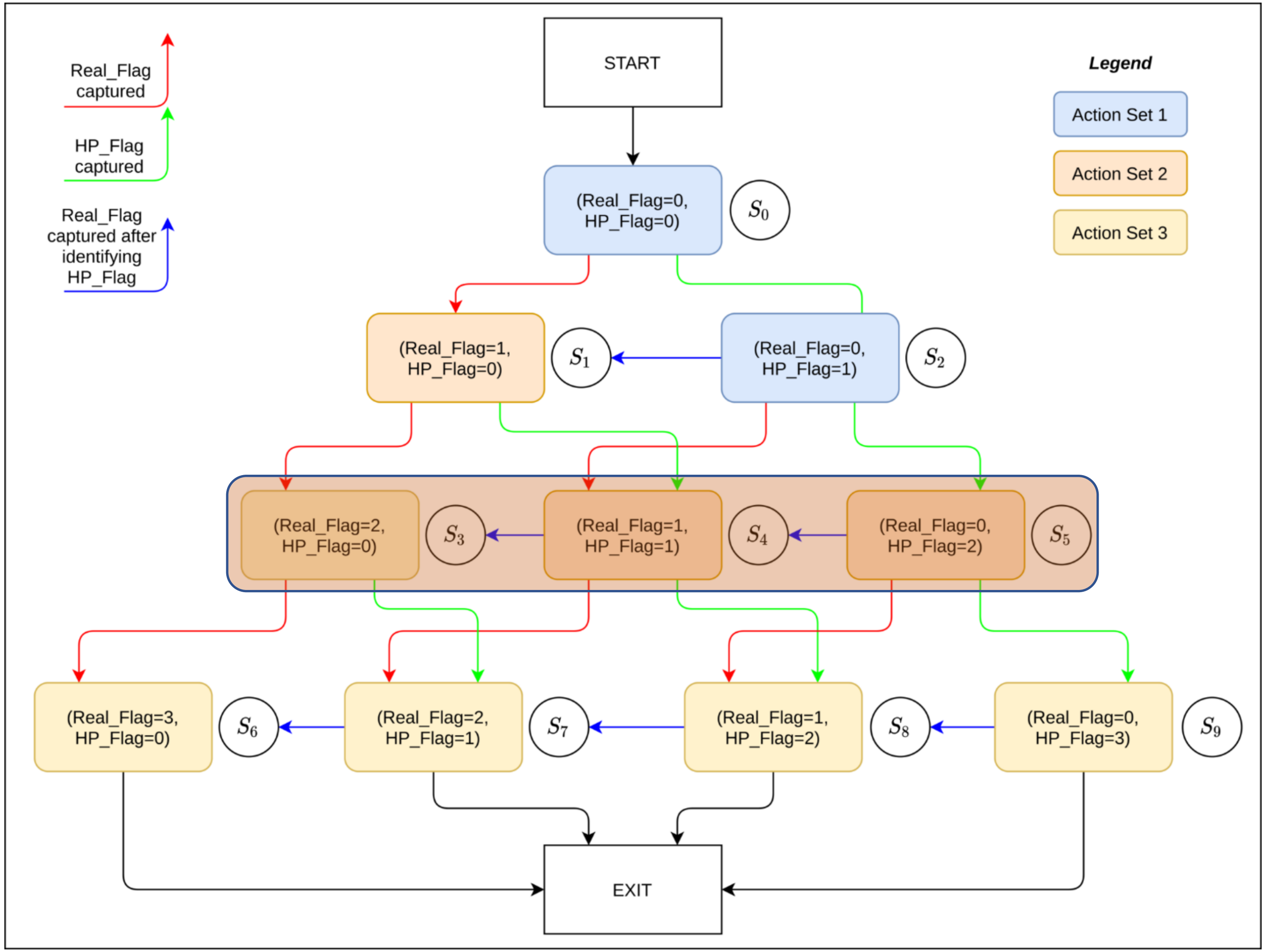


Which flag is it going to be?

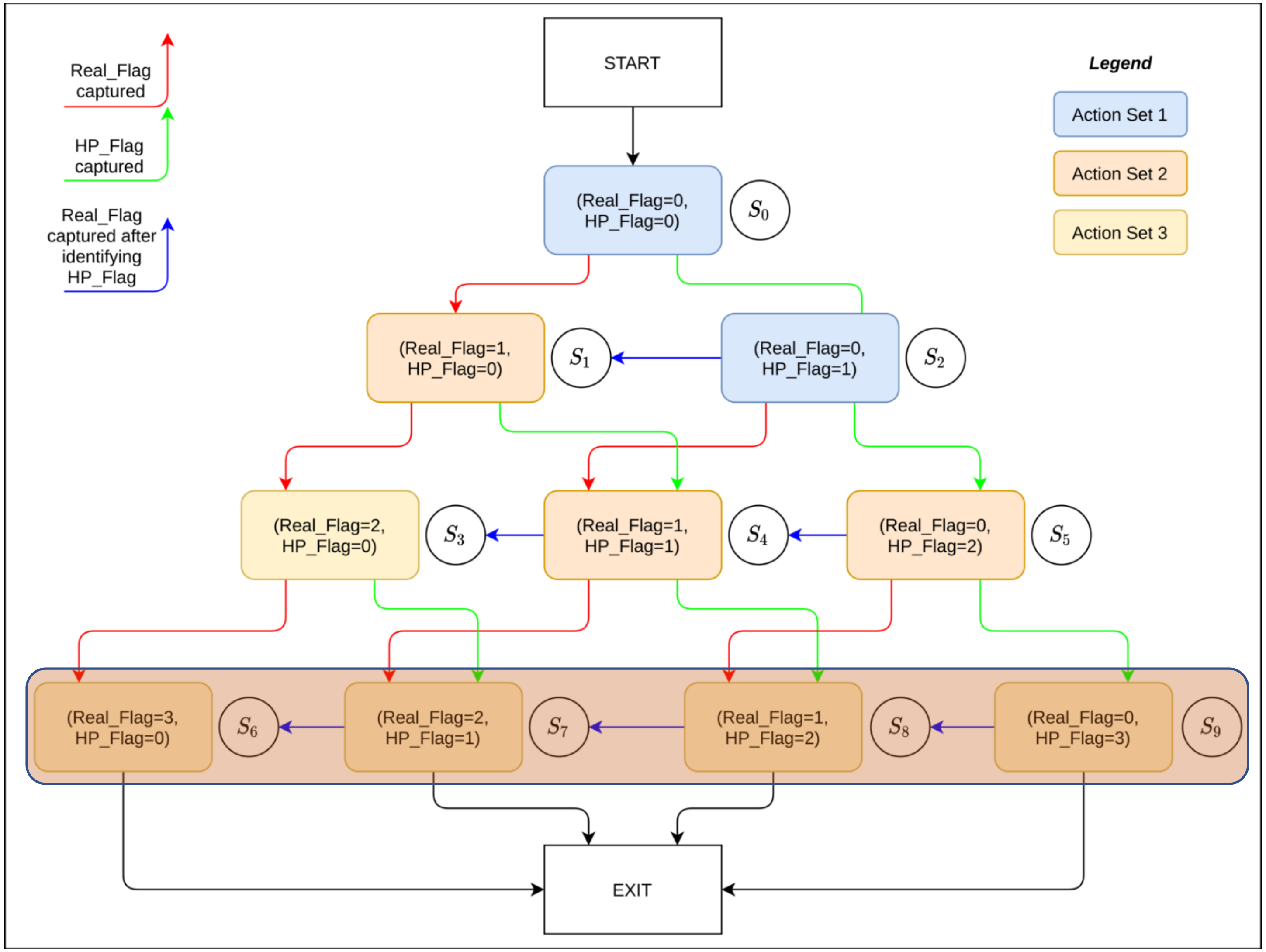
Attack Graph



Attack Graph



Attack Graph



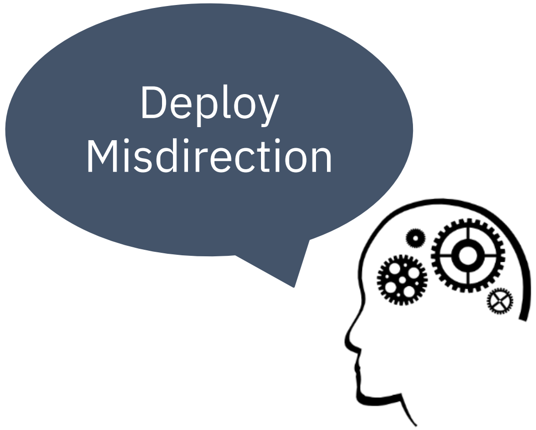
Agenda

- Reviewing Honey-Patching
- Setting up the user-study
- Deploying Honey-Patches as mitigations
- Modeling the adversary's attack graph
- Formulating the Markov Game model
- Computing the Bayesian Stackelberg Equilibrium

Formulating the Markov Game

Markov Game (Shapley 1953) for two players P_1 and P_2 can be defined by the tuple $(S, A_1, A_2, \tau, R, \gamma)$ where,

- $\mathbf{S = \{s_1, s_2, ..., s_k\}}$ represents a set of finite states of the game,

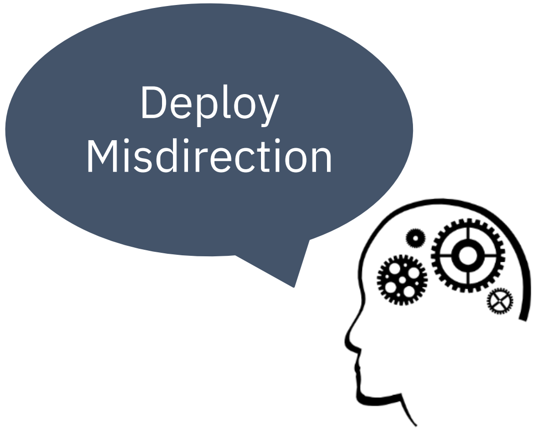


Game Theory Model

Formulating the Markov Game

Markov Game (Shapley 1953) for two players P_1 and P_2 can be defined by the tuple $(S, A_1, A_2, \tau, R, \gamma)$ where,

- $\mathbf{S = \{s_1, s_2, ..., s_k\}}$ represents a set of finite states of the game,
- $\mathbf{A1 = \{a^1{}_1, a^2{}_1, ..., a^m{}_1\}}$ represents the possible finite action sets for P_1,
- $\mathbf{A2 = \{a^1{}_2, a^2{}_2, ..., a^m{}_2\}}$ represents the possible finite action sets for P_2,

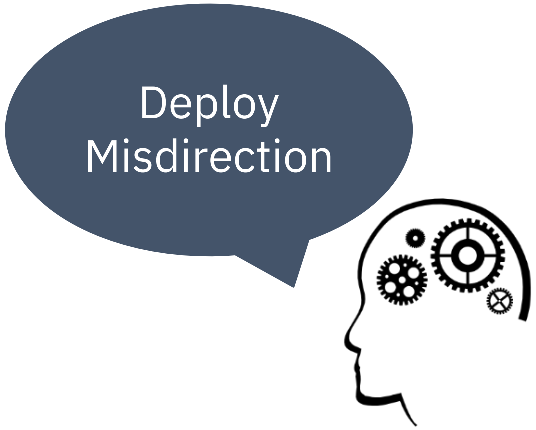


Game Theory Model

Formulating the Markov Game

Markov Game (Shapley 1953) for two players P_1 and P_2 can be defined by the tuple $(S, A_1, A_2, \tau, R, \gamma)$ where,

- $\mathbf{S = \{s_1, s_2, ..., s_k\}}$ represents a set of finite states of the game,
- $\mathbf{A1 = \{a^1{}_1, a^2{}_1 , ..., a^m{}_1\}}$ represents the possible finite action sets for P_1,
- $\mathbf{A2 = \{a^1{}_2, a^2{}_2 , ..., a^m{}_2\}}$ represents the possible finite action sets for P_2,
- $\boldsymbol{\tau}\ \mathbf{(s, a_1, a_2, s')}$ represents the transition probability matrix from a state $s \in S$ to $s' \in S$ when the P_1 and P_2 take actions a_1 and a_2 respectively,

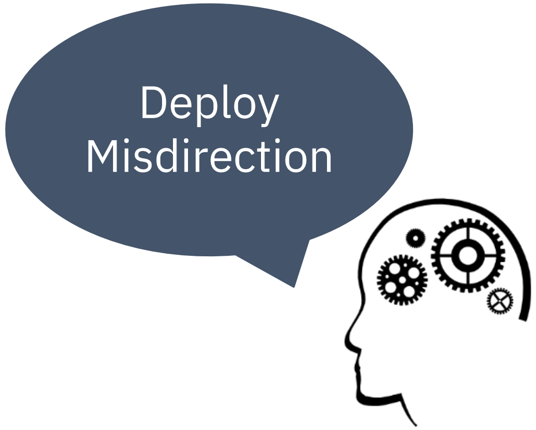


Game Theory Model

Formulating the Markov Game

Markov Game (Shapley 1953) for two players P_1 and P_2 can be defined by the tuple $(S, A_1, A_2, \tau, R, \gamma)$ where,

- $\mathbf{S = \{s_1, s_2, ..., s_k\}}$ represents a set of finite states of the game,
- $\mathbf{A1 = \{a^1{}_1, a^2{}_1 , ..., a^m{}_1\}}$ represents the possible finite action sets for P_1,
- $\mathbf{A2 = \{a^1{}_2, a^2{}_2 , \ldots, a^m{}_2\}}$ represents the possible finite action sets for P_2,
- $\boldsymbol{\tau}$ $\mathbf{(s, a_1, a_2, s')}$ represents the transition probability matrix from a state $s \in S$ to $s' \in S$ when the P_1 and P_2 take actions a_1 and a_2 respectively,
- $\mathbf{R^i (s, a_1, a_2)}$ denotes the utility or the rewards received by P_i in state s when P_1 and P_2 take actions a_1 and a_2 respectively,
- $\boldsymbol{\gamma} \rightarrow \mathbf{[0, 1)}$ is discount factor for future rewards.



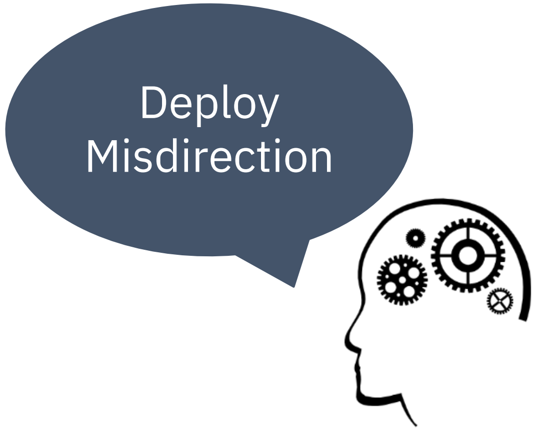
Game Theory Model

Formulating the Markov Game

Defender's actions ($A_{\mathcal{D}}$)

Attacker's actions ($A_{\mathcal{A}}$)

	no_mon	hp_1	hp_2	hp_3
no_op	0	-3	-3	-3
exp_1	-5.9	2.9	-8.9	-8.9
exp_2	-5.9	-8.9	2.9	-8.9
exp_3	-5.9	-8.9	-8.9	2.9



Game Theory Model

Chowdhary, Ankur, et al. "Markov game modeling of moving target defense for strategic detection of threats in cloud networks." *arXiv preprint arXiv:1812.09660* (2018).

Game Theoretic Modeling – Updating the Parameters

1. Naive Model

- Keeping all other parameters constant, we **randomly set the transition probabilities** τ for the game-theoretic model.

Game Theoretic Modeling – Updating the Parameters

1. Naive Model

- Keeping all other parameters constant, we **randomly set the transition probabilities** τ for the game-theoretic model.

2. Naive Model

- Keeping all other parameters constant, **a system expert sets the transition probabilities** τ for the game-theoretic model.

Game Theoretic Modeling – Updating the Parameters

1. Naive Model

- Keeping all other parameters constant, we **randomly set the transition probabilities τ** for the game-theoretic model.

2. Naive Model

- Keeping all other parameters constant, **a system expert sets the transition probabilities τ** for the game-theoretic model.

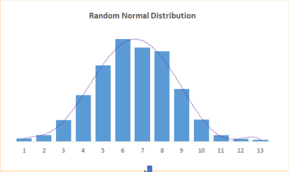
3. Updated Model

- Transition probability matrix **τ is calculated using the statistics** obtained from the iCTF user studies.

Game Theoretic Modeling – Updating the Parameters

1. Naive Model

- Keeping all other parameters constant, we **randomly set the transition probabilities τ** for the game-theoretic model.

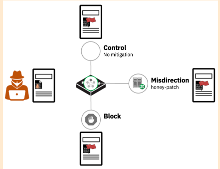


2. Naive Model

- Keeping all other parameters constant, **a system expert sets the transition probabilities τ** for the game-theoretic model.

3. Updated Model

- Transition probability matrix **τ is calculated using the statistics** obtained from the iCTF user studies.



τ

Agenda

- Reviewing Honey-Patching
- Setting up the user-study
- Deploying Honey-Patches as mitigations
- Modeling the adversary's attack graph
- Formulating the Markov Game model
- Computing the Bayesian Stackelberg Equilibrium

Finding the Optimal Defender Strategy

Algorithms:

- **MMP**: Min Max Pure Strategy
- **URS**: Uniform Random Strategy
- **OPT**: Optimal Mixed Strategy

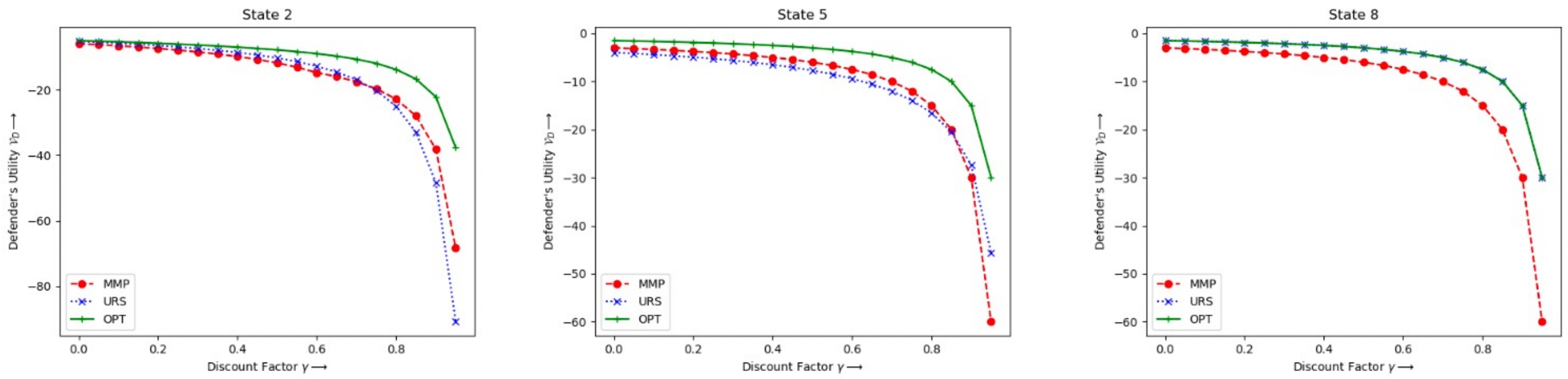


Figure: Defender’s payoffs for Naive Model - **randomly set transition probabilities** and uniform mitigation deployment costs.

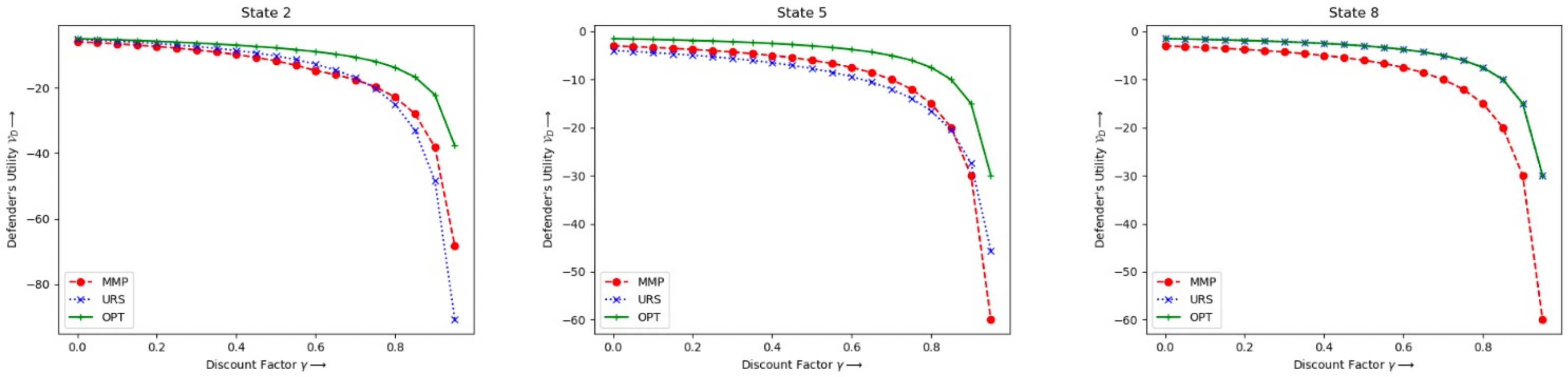
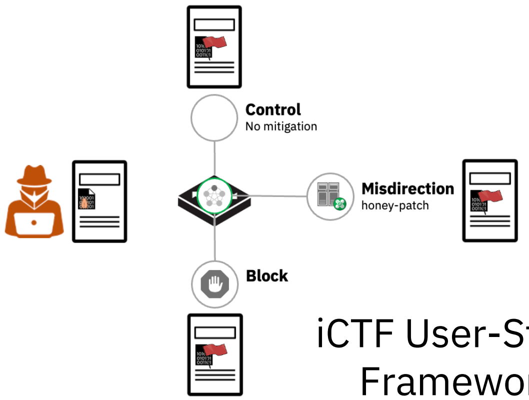


Figure: Defender’s payoffs for Naive Model – **system expert set transition probabilities** and uniform mitigation deployment costs.

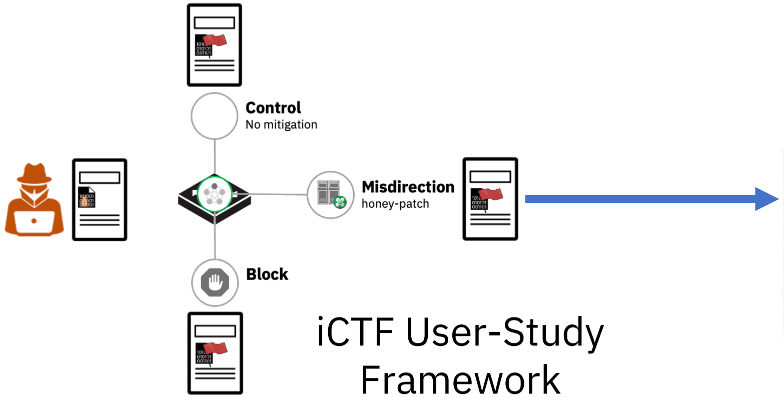
OPT does the best!

Updating the Model using iCTF User Studies



iCTF User-Study Framework

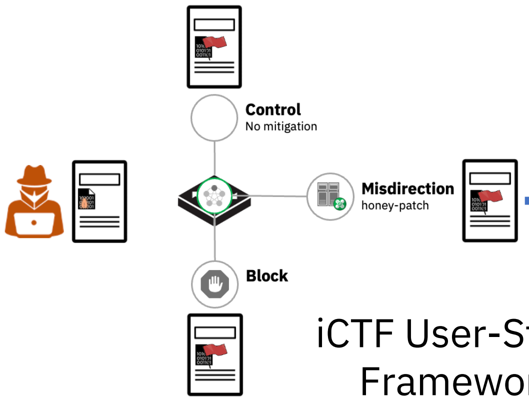
Updating the Model using iCTF User Studies



iCTF User-Study Framework

			Flag Captured w/ Defense Mechanism			
			None	Snort	Honeypatch	
Challenge	**Total Experiments**	**Timed Out**	**Real Flag**		**Real Flag**	**Honeypot Flag**
backup	18	0	6/6	6/6	0/6	6/6
sampleak	18	9	4/6	2/5	0/7	3/7
exploit-market	15	6	3/5	3/5	0/5	3/5

Updating the Model using iCTF User Studies

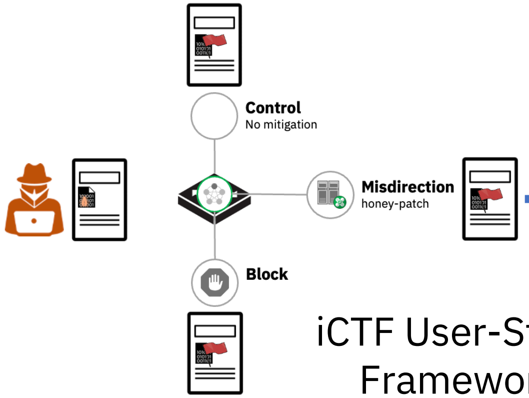


iCTF User-Study Framework

			Flag Captured w/ Defense Mechanism			
			None	Snort	Honeypatch	
Challenge	Total Experiments	Timed Out	Real Flag		Real Flag	Honeypot Flag
backup	18	0	6/6	6/6	0/6	6/6
sampleak	18	9	4/6	2/5	0/7	3/7
exploit-market	15	6	3/5	3/5	0/5	3/5

Observation	Vulnerable Application	Probability
Real flag captured if no honeypot	*backup*	1.0
	sampleak	0.43
	exploit-market	0.4
Trapped in honeypot	*backup*	1.0
	sampleak	0.5
	exploit-market	0.6

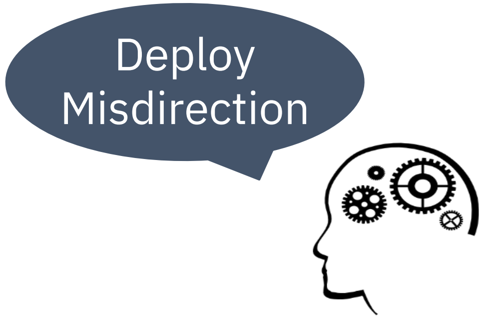
Updating the Model using iCTF User Studies



iCTF User-Study Framework

			Flag Captured w/ Defense Mechanism			
			None	Snort	Honeypatch	
Challenge	Total Experiments	Timed Out	Real Flag		Real Flag	Honeypot Flag
backup	18	0	6/6	6/6	0/6	6/6
sampleak	18	9	4/6	2/5	0/7	3/7
exploit-market	15	6	3/5	3/5	0/5	3/5

Observation	Vulnerable Application	Probability
Real flag captured if no honeypot	*backup*	1.0
	sampleak	0.43
	exploit-market	0.4
Trapped in honeypot	*backup*	1.0
	sampleak	0.5
	exploit-market	0.6



Game Theory Model

Comparing the Game Theoretic Model Case Studies

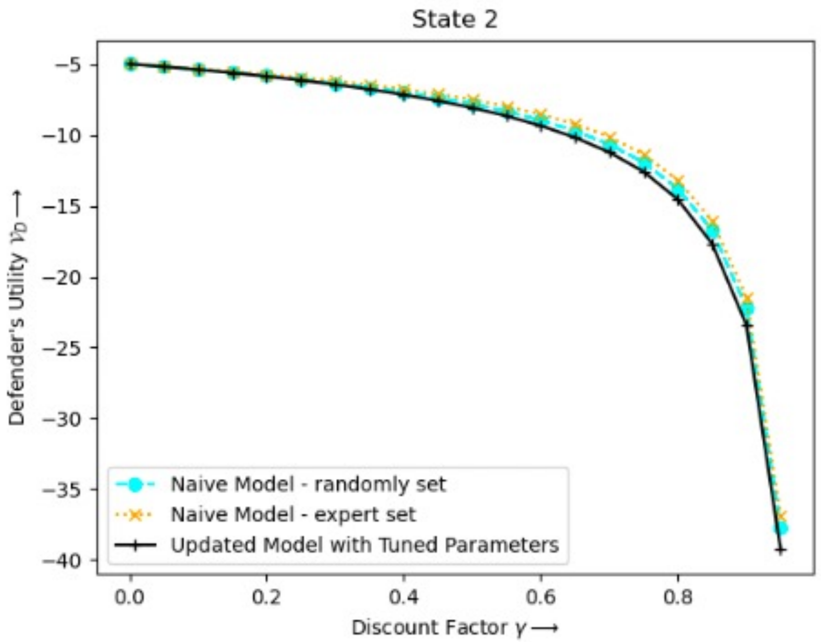


Figure: Defender's payoffs compared for the three models using **uniform mitigation deployment costs**.

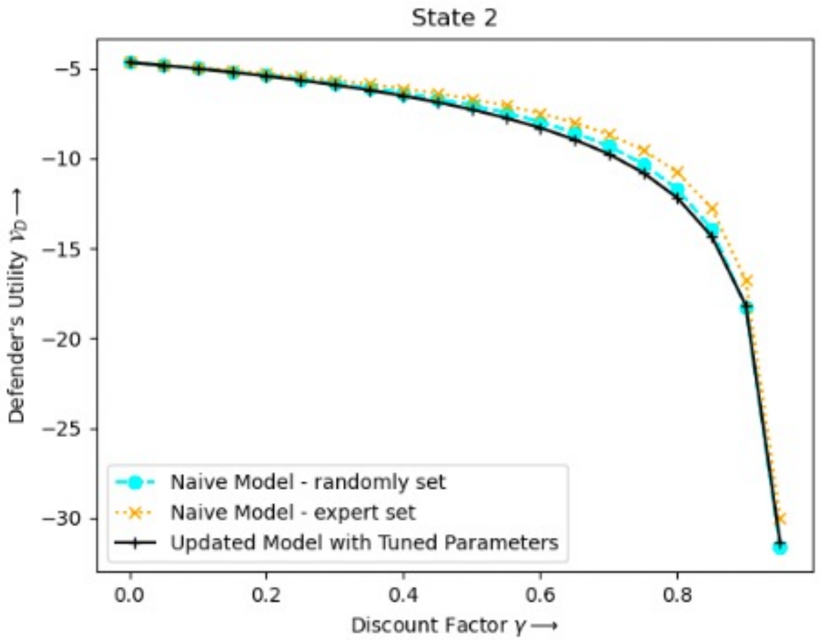


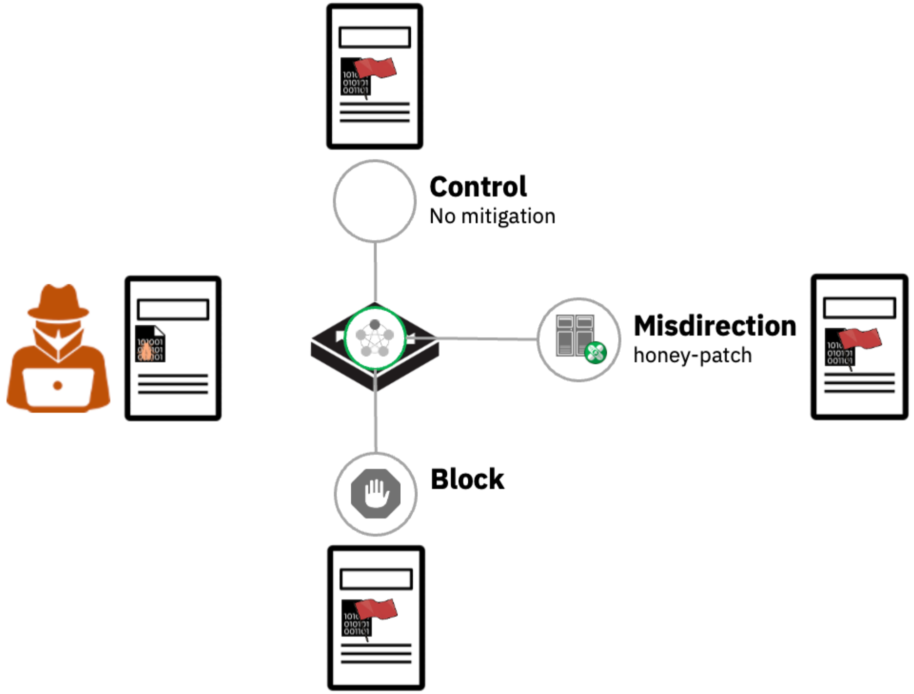
Figure: Defender's payoffs compared for the three models using **non-uniform mitigation deployment costs**.

Defender's returns:

Expert set (τ) **>** Randomly set (τ) **>** Computed from iCTF user studies (τ)

Evaluating the Hypothesis

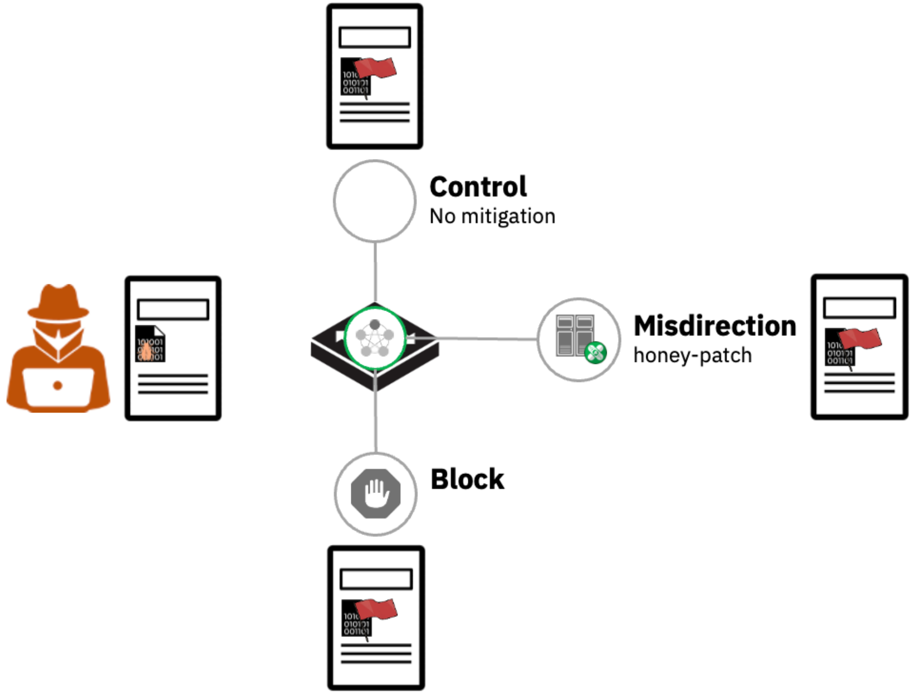
Hypothesis: Once trapped in a honeypot environment, the attacker *chooses to continue* with the existing strategy to exploit the remaining vulnerabilities.



Evaluating the Hypothesis

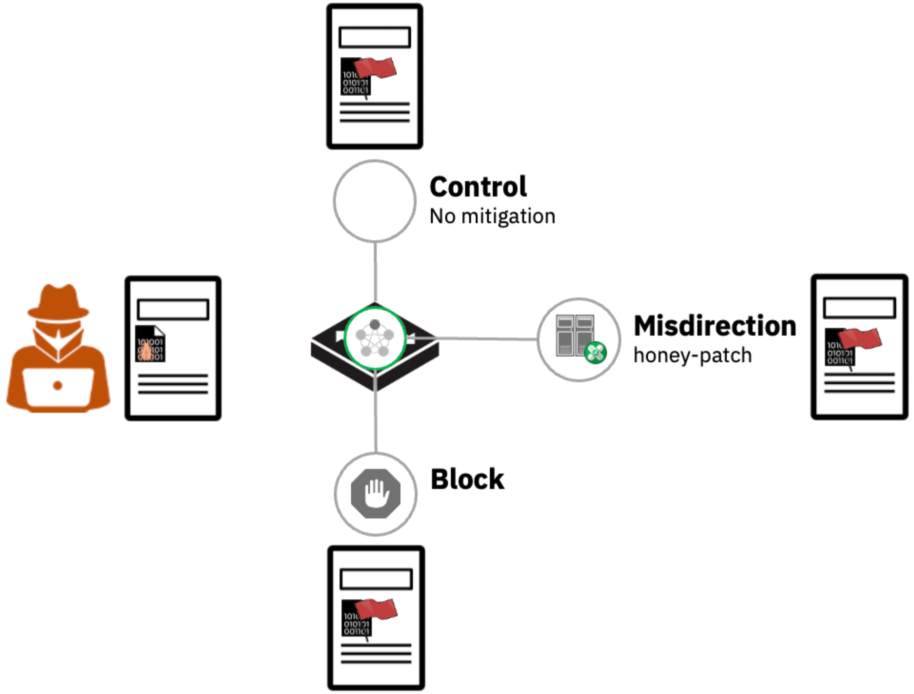
Hypothesis: Once trapped in a honeypot environment, the attacker *chooses to continue* with the existing strategy to exploit the remaining vulnerabilities.

Observation 1: None of the attackers received the observation of being trapped in a honey-pot, and thus continued with their existing strategy.



Evaluating the Hypothesis

Hypothesis: Once trapped in a honeypot environment, the attacker *chooses to continue* with the existing strategy to exploit the remaining vulnerabilities.

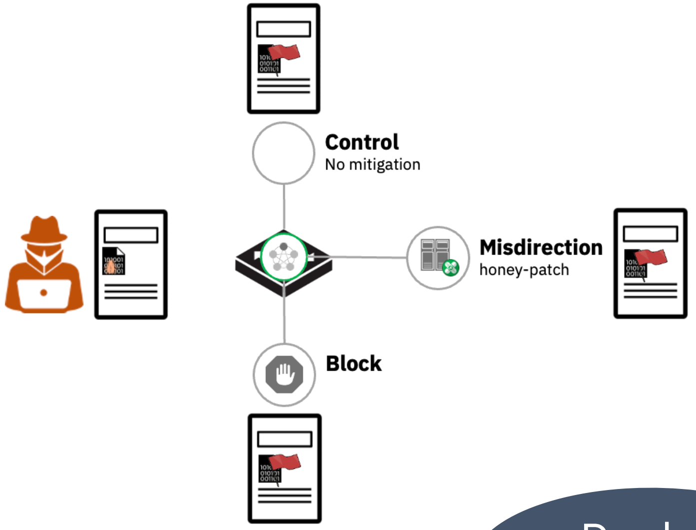


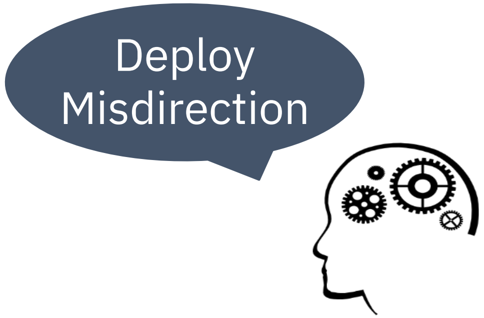
Observation 1: None of the attackers received the observation of being trapped in a honey-pot, and thus continued with their existing strategy.

Observation 2: Only in one instance, an attacker learns about the honeypot for sampleak, after the user study ends and the attacker is explicitly informed.

Conclusion

- Utility returns vary in the three models only for the initial 3 states of the game.

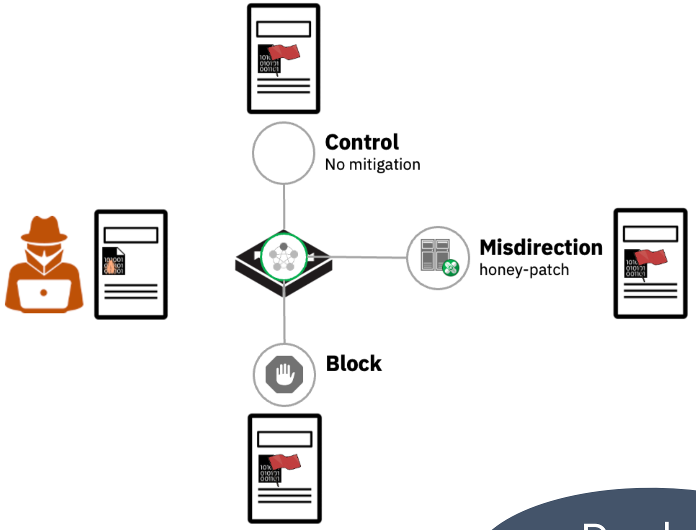


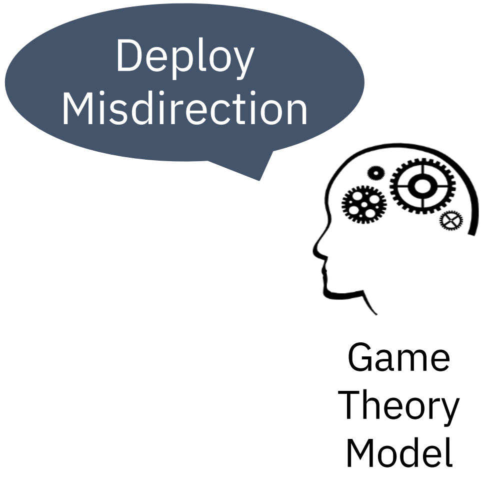


Game Theory Model

Conclusion

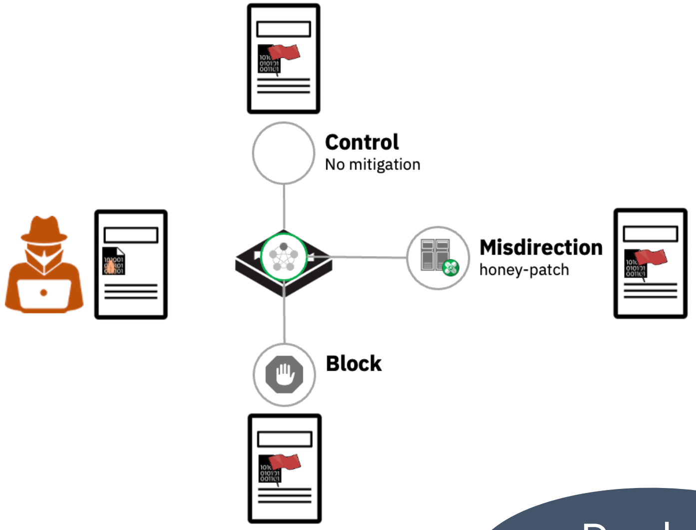
- Utility returns vary in the three models only for the initial 3 states of the game.
- Irrespective of parameter settings, all models recommend the defender to honey-patch the next vulnerable application.

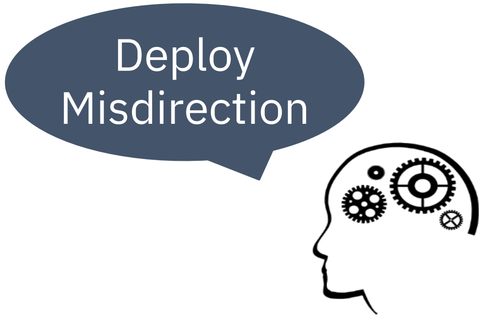




Conclusion

- Utility returns vary in the three models only for the initial 3 states of the game.
- Irrespective of parameter settings, all models recommend the defender to honey-patch the next vulnerable application.
- Model parameters set randomly or by expert may not imitate the true model representative of the real-world attack scenario.

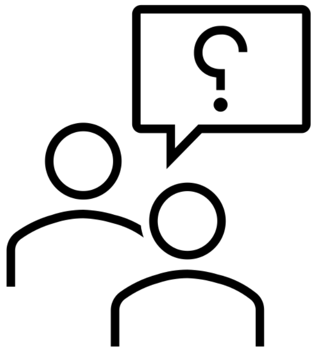




Game Theory Model

Summary

- Cybersecurity exercise helped gain insightful knowledge
- Closely analyzed interactions in deception-based setup
- Relaxed control over the different modalities
- Explore more ways to analyze attack behavior
- Obtain truer estimates for game-theoretic models



Thank You!